

**UTILITY
PATENT APPLICATION
TRANSMITTAL**

(Only for new nonprovisional applications under 37.1.53(b))

Attorney Docket No.	18617.0049
First Inventor	Robert Van Renesse et al.
Title	Method And System For Optimizing ...
Express Mail Label No.	EK310417718US

APPLICATION ELEMENTS

See MPEP chapter 600 concerning utility patent application contents

Assistant Commissioner for Patents
ADDRESS TO: Box Patent Application
Washington, D.C. 20231

1. ☒ Fee Transmittal Form
(Submit an original, and a duplicate for fee processing)
2. ☒ Applicant claims small entity status.
See CFR 1.27.
3. ☒ Specification [Total Pages / 31 /]
(preferred arrangement set forth below)
 - Descriptive title of the invention
 - Cross Reference to Related Applications
 - Statement Regarding Fed sponsored R&D
 - Reference to sequence listing, a table, or a computer program listing appendix
 - Background of the Invention
 - Brief Summary of the Invention
 - Brief Description of the Drawings (if filed)
 - Detailed Description
 - Claim(s)
 - Abstract of the Disclosure
4. ☒ Drawing(s) (35 USC 113) [Total Sheets / 3 /]
5. ☒ Oath or Declaration [Total Pages / 2 /]
 - a. ☐ Newly executed (original or copy)
 - b. ☒ Copy from a prior application (37 CFR 1.63(d))
(for continuation/divisional with Box 17 completed)
 - i. ☐ **DELETION OF INVENTOR(S)**
Signed statement attached deleting inventor(s) named in the prior application, see 37 CFR 1.63(d)(2) and 1.33(b).
6. ☐ Application Data Sheet. See 37 CFR 1.76

7. ☐ CD-ROM or CD-R in duplicate, large table or Computer Program (Appendix)
8. Nucleotide and/or Amino Acid Sequence Submission
(if applicable, all necessary)
 - a. ☐ Computer Readable Form (CRF)
 - b. ☐ Specification Sequence Listing on:
 - i. ☐ CD-ROM or CD-R (2 copies); or
 - ii. ☐ paper
 - c. ☐ Statements verifying identity of above copies

ACCOMPANYING APPLICATION PARTS

9. ☐ Assignment Papers (cover sheet & document(s))
10. ☐ 37 CFR 3.73(b) Statement ☐ Power of Attorney
(when there is an assignee)
11. ☐ English Translation Document (if applicable)
12. ☐ Information Disclosure Statement (IDS)/PTO-1449 ☐ Copies of IDS Citations
13. ☒ Preliminary Amendment
14. ☒ Return Receipt Postcard (MPEP 503)
(Should be specifically itemized)
15. ☐ Certified Copy of Priority Document(s)
(if foreign priority is claimed)
16. ☒ Other: fee transmittal; check for \$463

17. If a CONTINUING APPLICATION, check appropriate box and supply the requisite information below and in a preliminary amendment, or in an Application Data Sheet under 37 CFR 1.76:☒ Continuation ☐ Divisional ☐ Continuation-in-part (CIP)

of the prior application No: 09 / 143,620

Prior application information: Examiner: B. Pham

Group/Art Unit: 2731

For CONTINUATION OR DIVISIONAL APPS only: The entire disclosure of the prior application, from which an oath or declaration is supplied under Box 5b, is considered a part of the disclosure of the accompanying continuation or divisional application and is hereby incorporated by reference. The incorporation can only be relied upon when a portion has been inadvertently omitted from the submitted application parts.

18. CORRESPONDENCE ADDRESS

NAME	Martin G. Linihan				
	Hodgson, Russ, Andrews, Woods & Goodyear, LLP				
ADDRESS	One M&T Plaza, Suite 2000				
CITY	Buffalo	STATE	New York	ZIP CODE	14203-2391
COUNTRY	United States of America	TELEPHONE	(716) 856-4000	FAX	(716) 849-0349

"Express Mail" Mailing Label Number EK310417718US

Date of Deposit October 13, 2000

I hereby Certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231.

Martin G. Linihan
Name

Signature

FEE TRANSMITTAL for FY 2000

Patent Fees are subject to annual revision.

Application Number

Filing Date

October 13, 2000

First Named Inventor

Robert Van Renesse et al.

Examiner Name

Group/Art Unit

TOTAL AMOUNT OF PAYMENT

(\$463.00)

Attorney Docket Number

18617.0049

1. ☒ The Commissioner is hereby authorized to charge indicated fees and credit any overpayments to:

3. ADDITIONAL FEES

Deposit Account Number: 08-2442
Deposit Account Name:
Hodgson, Russ, Andrews, Woods & Goodyear, LLP

Large
Fee
CodeEntity
Fee
(\$)Small
Fee
CodeEntity
Fee
(\$)

Fee Description

Fee
Paid

☒ Charge Any Additional Fee Required Under 37 CFR 1.16 and 1.17

105

130

205

65

Surcharge - late filing fee or oath

\$

☒ Applicant claims small entity status. See 37 CFR 1.27.

127

50

227

25

Surcharge - late provisional filing fee or cover sheet

\$

2. ☒ Payment Enclosed:

☒ Check ☐ Credit Card ☐ Money Order ☐ Other

139

130

139

130

Non-English specification

\$

FEE CALCULATION

147

2,520

147

2,520

For filing a request for reexamination

\$

1. FILING FEE

Large Entity Small Entity

112

920*

112

920*

Requesting Publication of SIR prior to Examiner Action

\$

Fee Code (\$)

113

1,840*

113

1,840*

Requesting Publication of SIR after Examiner Action

\$

107 710 201 355 Utility filing fee \$355

115

110

215

55

Extension for reply within first month

\$

106 320 206 160 Design filing fee \$

116

390

216

195

Extension for reply within second month

\$

107 490 207 245 Plant filing fee \$

117

890

217

445

Extension for reply within third month

\$

108 710 208 355 Reissue filing fee \$

118

1,390

218

695

Extension for reply within fourth month

\$

114 150 214 75 Provisional filing fee \$

128

1,890

228

945

Extension for reply within fifth month

\$

SUBTOTAL (1) \$355

119

310

219

155

Notice of Appeal

\$

2. EXTRA CLAIM FEES

Extra Fee from
Claims belowFee
Paid

120

310

220

155

Filing a brief in support of an appeal

\$

Total Claims /32 / -20** = /12 / x /9 \$108

121

270

221

135

Request for oral hearing

\$

Independent Claims /2 / - 3** = /0 / x /40 / \$0

138

1,510

138

1,510

Petition to institute a public use proceeding

\$

Multiple dependent / / x / / = \$

140

110

240

55

Petition to revive - unavoidable

\$

Large Entity Small Entity

141

1,240

241

620

Petition to revive - unintentional

\$

Fee Code (\$)

142

1,240

242

620

10 advance copies
Utility issue fee (or reissue)

\$

103 18 203 9 Claims in excess of 20

143

440

243

220

Design issue fee

\$

102 80 202 40 Independent claims in excess of 3

144

600

244

300

Plant issue fee

\$

104 270 204 135 Multiple dependent claim if not paid

122

130

122

130

Petitions to the Commissioner

\$

109 80 209 40 **Reissue independent claims over original patent

123

50

123

50

Petitions related to provisional applications

\$

110 18 210 9 **Reissue claims in excess of 20 and over original patent

126

240

126

240

Submission of Information Disclosure Statement

\$

SUBTOTAL (2) \$108

581

40

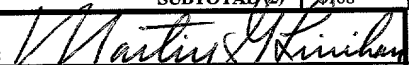
581

40

Recording each patent assignment per property (times number of properties)

\$

SIGNATURE:



146

710

246

355

Filing a submission after final rejection(37 CFR 1.129(a))

\$

Martin G. Linihan

Reg. No. 24,926

149

710

249

355

For each additional invention to be examined
(37 CFR 1.129(b))

\$

DATE: October 13, 2000

Telephone: (716) 848-1367

*Reduced by basic filing fee paid

SUBTOTAL (3) \$0

Express Mail® Mailing Label Number EK310417718US

Date of Deposit: October 13, 2000

I hereby Certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231.

Martin G. Linihan

Name


Signature

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of

Robert Van Renesse et al.

A Continuation of
Serial No. 09/143,620
Filed: October 28, 1998

For: Method And System For Optimizing Layered
Communication Protocols

PRELIMINARY AMENDMENT

Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

Upon granting of a serial number and filing date to the
above-identified continuation application, please amend the
application as follows:

In the Claims:

Please cancel claim 1.

Respectfully submitted,

By Martin G. Linihan
Martin G. Linihan
Reg. No. 24,926

Hodgson, Russ, Andrews
Woods & Goodyear, LLP
1800 One M&T Plaza
Buffalo, New York 14203
Tel: 716-848-1367
October 13, 2000
BFLDOCS:451526_1 (9_##01)

DOE T.O.T. 65428950

another, application designers can select a combination of protocols most suited to their expected work load. In addition, systems such as Ensemble support changing protocol stacks underneath executing applications, so
5 the application can tune its protocol stack to its changing work load.

Unfortunately, the convenience of having a stack of protocols is often overshadowed by the problem that
10 layering produces a lot of overhead which, in turn, increases delays in communication. Extensively layered group communication systems where high-level protocols are often implemented by 10 or more protocol layers greatly reduce design complexity of a communication
15 network. On the other hand, extensive layering often leads to serious performance inefficiencies.

The disadvantages of layered systems leading to performance inefficiencies consist primarily of overhead, both in computation and in message headers,
20 caused by the abstraction barriers between layers. Because a message often have to pass through as many as 10 or more protocol layers on its way from a host to the network and from the network to a host, the overhead produced by the boundaries between the layers is often
25 more than the actual computation being done. Different system have reported overheads for crossing layers of up to 50 μ s. Therefore, it is highly desirable to mitigate the disadvantages and to develop techniques that reduce delays by improving performance of layered protocols.

30 Several methods have been suggested to improve performance of layered communication protocols. One of the methods is described by Robbert van Renesse in the article "Masking the Overhead of Protocol Layering", Proc. of the Proceedings of the 1996 ACM SIGCOMM

090949 0400

The described protocol accelerator optimization model successfully reduces communication latency, but
30 does not decrease actual computation and layering overhead. It would also be desirable to optimize a larger class of communication protocols, including outing and total ordering protocols. Moreover, the protocol accelerator approach requires structural

The described protocol accelerator optimization model successfully reduces communication latency, but
30 does not decrease actual computation and layering overhead. It would also be desirable to optimize a larger class of communication protocols, including outing and total ordering protocols. Moreover, the protocol accelerator approach requires structural

modifications to protocols that are effectively annotations. It would be desirable to employ such optimization that calls for significantly less annotation.

5 Other work on protocol optimization has been done on Integrated Layer Processing (ILP) in "Analysis of Techniques to Improve Protocol Processing latency; in *Proc. of the Proceedings of the 1996 ACM SIGCOMM Conference, Stanford, September 1996,*" and "RPC in the
10 x-Kernel: Evaluating New Design Techniques; In *Proc. of the Fourteenth ACM SYMP. on Operating Systems Principles*, pages 91-101, Asheville, NC, December 1993.. ILP encompasses optimizations on multiple protocol layers. Much of the ILP tends to focus on integrating
15 data manipulations across protocol layers, but not on optimizing control operations and message header compression. On the other hand, ILP advantageously compiles iteration in checksums, presentation formatting, and encryption from multiple protocol layers
20 into a single loop to minimize memory references. Currently, none of the Ensemble protocols touch the application portion of messages. It would be desirable to provide improved optimization techniques incorporating the advantages of already developed
25 optimizations and focusing on such aspects of protocol execution that are compatible with and orthogonal to the existing optimization methods.

The above-described disadvantages of the previously developed optimization methods make it desirable to
30 develop compilation techniques which make layered protocols execute as fast as non-layered protocols without giving up the advantages of using modular, layered protocol suites.

It is therefore an object of the present invention to provide a system and method which decreases actual computation and layering overhead in addition to latency and to provide optimization techniques applicable to a larger class of protocols.

It is yet another object of the present invention to provide optimized protocols of high performance which are easy to use. Normally, the protocol optimizations are made after-the-fact to already working protocols. This means that protocols are designed largely without optimization issues in mind. In the present invention optimizations require almost no additional programming, only a minimal amount of annotation of the protocol

layers is necessary (the annotation consists of marking the start and end of the common paths of the source code). Therefore, optimizations can call for annotating only small portions of the protocols which belong to the common path, reducing the complexity of the optimization techniques. In addition, the optimizations of the present invention place few limitations on the execution model of the protocol layers.

It is also an object of the present invention to be able to apply the current state of verification technology to small, layered protocols which are just within the range of current verification technologies, whereas large, monolithic protocols are certainly outside this range.

15

BRIEF DESCRIPTION OF THE DRAWING FIGURES

Figure 1 is a schematic comparison of protocol layers and event traces.

20

Figure 2 is a block diagram illustrating elements of a layering model.

Figure 3 is a block diagram illustrating event traces, trace handlers, and trace conditions.

25

Figure 4 is a block diagram illustrating a complex, non-linear trace in a routing protocol stack.

Figure 5 is a chart representing performance comparison for various protocol stacks.

30

Figure 6 is an illustration of a round-trip latency time line between two processes.

1. Layering Model

10

15

20

25

30

layer to the lower layer, and another for the other direction.

5. An Application 28 and a Network 30. The
5 application communicates with the top of the protocol
stack: messages are sent by introducing *send* events into
the top of the stack, and are received by through
receive events that are emitted from the top. The
network communicates with the bottom of the protocol
10 stack. *Send* events that emerge from the bottom layer of
the protocol stack cause a message to be transmitted
over the underlying network. *Receive* events cause the
messages to be inserted into the bottom of the stack of
the destination.

15
6. A Scheduler 32 determines the order of execution of
events in a protocol stack. The scheduler must ensure
that events are passed between adjacent layers in the
first-in-first-out order and that any particular
20 protocol layer is executing at most one event at a time.
Also, all events must eventually be scheduled.

7. An Event trace 34 is a sequence of operations in a
protocols stack. In particular, the term "event trace"
25 is used to refer to the traces that arise in the normal
case. Event trace 34 begins with the introduction of
single event into protocol stack 26. The trace
continues through the protocol layers, where other
events may be spawned either up or down. In many cases
30 even trace 34 may be scheduled in various ways. It is
assumed that a particular schedule is chosen for a
particular trace.

8. A trace condition 40 is a condition under which a particular event trace will be executed. The condition usually consists of a predicate on the local states of the layers in a protocol stack and on an event about to be introduced to the protocol stack. If the predicate is true then the layers will execute the corresponding trace as a result of the event.

9. A Trace handler 36 comprises the sequence of operations executed in a particular event trace. If the trace condition holds for trace handler 36 then executing the handler will be equivalent to executing the operations along the common path within the protocol layers.

10. Complex event traces are nonlinear with event traces at 34. Many protocol stacks have event traces that are not linear. Nonlinear traces have multiple events that are passed in both directions through the protocol stack. Nonlinear event traces are important, because they occur in many protocol stacks, so without support for such traces these stacks could not be optimized. Examples of such protocols include token-based total ordering protocols, broadcast stability detection protocols, and hierarchical broadcast protocols.

In a simple case of sending a message from a sending host to a destination host, application 28 inserts a *send* event into the top of protocol stack 26. The event is passed to the topmost protocol layer, such as layer 24 in Fig. 2, which executes its handler on the event. The layer then updates its state and emits zero or more events. In a simple scenario, the same event gets passed from one layer to the next all the way to

the bottom of the protocol stack. When the event emerges from the stack, network 30 transmits the message. The destination host inserts a *receive* event into the bottom of the protocol stack. Again, in a
5 simple scenario the event is repeatedly passed up to the top of the protocol stack and is handed to the application. In more complex situations, a layer can generate multiple events when it processes an event. For instance, a reliable communication layer may both
10 pass a *receive* event it receives to the layer above it, and pass an *acknowledgment* event to the layer below.

This model is flexible in that scheduler 32 has few restrictions on the scheduling. For example, the model admits a concurrent scheduler where individual layers
15 execute events in parallel.

The optimizations of the present invention were implemented as a part of the Ensemble communication system, which is described below. For an application builder, Ensemble provides a library of protocols that
20 can be used for quickly building complex distributed applications. An application registers 10 or so event handlers with Ensemble, and then the Ensemble protocols handle the details of reliably sending and receiving messages, transferring state, detecting failures, and
25 managing reconfigurations in the system. For a distributed systems user, Ensemble is a highly modular and reconfigurable toolkit. The high-level protocols provided to applications comprise stacks of small protocol layers. Each of these protocol layers
30 implements several simple properties: providing sets of high-level properties such as, for example, total ordering, security and virtual synchrony. Individual protocol layers can be modified or rebuilt to test with new properties or change the performance characteristics

of the system, thus making Ensemble a very flexible platform for developing and testing optimizations to layered protocols.

As illustrated in Fig.3, original protocol stack 26 is embedded in an optimized protocol stack 38 in which the events that satisfy trace conditions 40 are intercepted and execute through heavily optimized trace handlers 36. Pictured in Fig. 3 is the original execution of the event trace and the interception of that trace with a trace handler. Multiple traces are optimized with each trace having its own trace condition and handler. In addition the present invention contemplates traces starting both at the bottom and the top of the protocol stack.

II. Common Paths in Layered Systems.

Common execution paths of events passed between the protocol layers in a communication system is the first step in the optimization method of the present invention. The old adage, "90% of the time is spent in 10% of a program," says that most programs have common paths, even though it is often not easy to find the common path. However, carefully designed systems often do a good job in exposing this path. In layered communication systems, the designer is often able to easily identify the common execution path for individual protocols, so these common paths can be composed together to arrive at global sequences of operations. It is these sequences, or event traces, that serve as the basic unit of execution and optimization. For each event trace, a condition which must hold for the trace to be enabled is identified, together with a handler that executes all of the operations in the trace.

As an example, a type of event trace that occurs in many protocol stacks is considered. When there are no abnormalities in the system, sending a message through a protocol stack often involves passing a *send* event directly through the protocol stack from one layer to the next. If messages are delivered reliably and in correct order by the underlying transport, then the actions at the receiving side involve a *receive* event filtering directly up from the network, through the layers, to the application. Such an event trace is depicted in Fig. 3 at 34. Both the *send* and receive event traces are called linear traces because (1) they involve only single events, and (2) they move in a single direction either from network 30 to application 28 or vice versa through the protocol stacks.

For example, a *hierarchical routing protocol* is a protocol in which a broadcast to many destinations is implemented through a spanning tree of the destinations. As illustrated in Fig. 4, a message is received from the network and passed to the routing layer. The routing layer forwards a copy down to the next destination and passes a copy to the network. The initiator sends the message to its neighbors in the tree, who then forward it to their children, and so on until it gets to the leaves of the tree which do not forward the message. Some of the traces in a hierarchical routing protocol would include the following steps, the first two of which are linear and the last step is non-linear:

1. Sending a message is a linear trace down through the protocol stack.

- 5
10

15

20

25

30

has slowed the protocol down a little. If a trace condition holds, then the normal event execution is intercepted and instead the trace handler is executed. The performance improvement then depends on the
5 percentage of events for which the trace condition is enabled, the overhead of checking the conditions, and how much faster the trace handler is.

The use of a trace handler assumes that there are no events pending in any of the intervening event
10 queues. If there were a pending event, the trace handler would violate the model because the events in the trace would be executed out of order with regard to the previously queued event. The solution to this problem relies on the flexibility of the layering model,
15 and works by using a special event scheduler that executes all pending events to completion before attempting to bypass a protocol stack, ensuring that there are no intervening events.

The transformation of the protocol stack maintains
20 correctness of the protocols because trace handlers execute exactly the same operations as could occur in the normal operation of the protocol layers, ensuring, therefore, the soundness of the transformation. If the original protocols are correct, then the trace protocols
25 are correct as well.

30 12. Optimizing Event Traces

After event traces are determined and common paths of execution based on the event traces are identified the event traces are then optimized. The optimization

techniques are divided into three classes: the first class of the techniques improve the speed of the computation; the second class compresses the size of message headers; and the third class reorders operations to improve communication latency without affecting the amount of computation.

a. Optimizing Computation

The first class of optimizations comprises optimization that improve the performance of the computation in event handlers. The general approach used by each optimizations is to carry out a set of transformations to the protocol stack so that traditional compilation techniques can be effectively applied.

The first step in optimizing computation extracts the source code corresponding to the trace condition and trace handler from the protocol layers. At this step it is convenient to break the operations of a stack into two types: protocol and layering operations. Protocol operations are those that are directly related to implementing a protocol, including operations such as message manipulations and state updates. Layering operations are those that result from the use of layered protocols, including but not limited to the costs of scheduling the event queues and the function call overhead from all the layers' event handlers. Layering operations are not strictly necessary because they are not parts of the protocols. Given an event trace and annotated protocol layers, annotations are used to textually extract the protocol operations for the trace from each layer.

5 information between protocol layers in a stack. These records contain temporary information about a message, which information follows the message through the layers and event queues. Each event must be allocated, initialized, and later released. It is not necessary to
10 use events explicitly, because event traces encompass the life of the initial event and all spawned events. Therefore, the contents of the event record can instead be kept in local variables within the trace handler. Compilers are often able to place such variables in
15 registers.

20 Normally, code explosion is an important concern when
inlining functions. However, the code explosion is not
an issue in this case, because there is only a small
number of trace handlers which are normally not too
large: the inlining is focussed on a small part of the
25 system so the code explosion will not be large.
Additionally, the functions called from trace handlers
are normally simple operations on abstract data types,
such as adding or removing messages from buffers. These
functions are not recursive and do not call many other
30 nested functions, so fully inlining them will typically
add only a fixed amount of code.

The fourth step is to apply traditional optimizations to the trace handlers. This operation proves to be very effective, because the previous passes

5 marks an event record's field with some flag to cause an operation to happen at another layer, the flap can be propagated through the trace handler so that the flap is never set at the first layer or checked at the second layer.

B. Compressing Protocol Headers.

1. Addressing headers are the headers used for routing messages, including addresses and other identifiers. They are treated opaquely: i.e., protocols are only interested in testing these headers for equality. Such headers are compressed through so-called path or connection identifiers, as described below.

2. Constant headers include headers that are one of several enumerated constant values and specify the "type" of the message. For instance, a reliable transmission protocol may mark messages as being "data" or "acknowledgments" with a constant header, and from this making the receiver knows how to treat the message.

These headers are compressed by our approach when they appear in the common path.

3. Non-constant headers include any other headers,
5 such as sequence numbers or headers used in negotiating reconfigurations. The non-constant headers are not compressed.

10 The above-described header compression optimizations are based on the use of connection identifiers, such as the ones described in U.S. patent application Serial No. 09/094,204, which is incorporated herein by reference. Connection identifiers are tuples containing addressing headers which do not change very
15 often. All the information in these tuples are hashed into 32-bit values which are then used along with hash tables to route messages to the protocol stacks. MD5 (a cryptographic one way hash function) is used to make hashing collisions very unlikely and other well-known
20 techniques can be used to protect against collisions when they occur. The use of connection identifiers compresses many addressing headers into a single small value. As a result, all subsequent messages benefit from such compression. Although the main goal of header
25 compression is to improve bandwidth efficiency, small headers also contribute to improved performance in transmitting the messages on the underlying network and in the protocols themselves because less data is being moved around.

30 In the present invention the concept of connection identifiers is extended to contain an additional field called the "multiplexing index." This field is used to multiplex several virtual channels over a single channel. Such use of connection identifiers allows

5

10

30

out-of-order messages may not be supported by trace handlers). Such a message must be passed to the normal execution of the protocol even though the message is not in the normal format. The second problem arises when a trace handler inserts a message into a buffer and a protocol layer later accesses the message. The solution to both problems lies in reformatting such messages. The messages are reformatted by functions which regenerate constant fields and move variable fields to their normal location in the messages. These reformatting functions can be generated automatically. To solve the first problem, the message is reformatted before being passed to the normal protocol stack. The protocol layers get the message as though it were delivered in the standard format.

In order to manage buffers containing messages in different formats, each message is marked as normal or compressed. Compressed messages are buffered along with their reformatting function. When a protocol accesses a compressed message, it first calls the function to reformat the message. For most protocols, normally a message is buffered and later released without further accesses by protocols. Reformatting is efficient in these cases, because messages are buffered in compressed form and, so no additional operations are carried out on the message. Handling the buffers requires some modification of the protocol layers. The modification is required only in the layers with message buffers, and in such layers the modification is usually very simple. The reformatting function needs to be stored with compressed messages, but the cost of storage is offset by the decreased size of the messages.

C. Delayed Processing

5
10

15

15

20
25
30

5

10

15

20

25

30

path by delaying operations. All measurements were made on Sparcstation 20s with 4 byte messages. Measurements were gathered for three protocol stacks: the non-optimized protocols, the optimized protocols entirely in ML, and the optimized protocols where the trace conditions and handlers have been rewritten in C. As shown in Fig. 5, the C version of the protocol stacks has approximately $5\mu s$ of overhead in the code-latency from parts of the Ensemble infrastructure that are in ML. This result can be further optimized by rewriting this significant infrastructure in C. There are no delayed operations in the non-optimized protocol stack.

The time line for the latency corresponding to one round-trip of the C protocol is depicted in Fig. 6. In this test two Sparcstation 20s are communicating over an ATM network using U-net which has one-way latencies of $35\mu s$. As shown in Fig. 6, at $0\mu s$ process A received a message from process B off the network. $26\mu s$ later the application received the message and the next message was sent on the network. At $61\mu s$, process B received the message and sent the next message at $87\mu s$. Process A completed its delayed updates by time $62\mu s$. The total round-trip time was $122\mu s$, of which Ensemble contributed $52\mu s$.

It is important to note that since the time the test results represented in Fig. 5 were obtained, significance improvements in ML compilers made it possible to achieve the performance of the optimized pure ML protocol stack similar to that of the C protocol.

It is therefore apparent that the present invention accomplishes its intended objects. While embodiments of the present invention have been described in detail, that is for the purpose of illustration, not limitation.

1. A method of improving performance efficiency in a communication system having a plurality of layered protocols by decreasing actual computation, communication latency and layering overhead, the method comprising the steps of:

determining at least one common execution path in the communication system by identifying a common sequence of operations occurring in the plurality of layered protocols and identifying at least one condition allowing an event to be executed along the common execution path; and

optimizing the speed of execution of the event along the common execution path by extracting the source code corresponding to the condition and the common sequence of operations, eliminating intermediate data structures and inlining functions called from the common sequence of operations.

2. A method of improving performance efficiency in a communication system having a plurality of layered protocols in a protocol stack, the method comprising the steps of:

determining at least one common execution path in the protocol stack by identifying a common sequence of operations occurring in the plurality of layered protocols;

identifying at least one condition allowing an event to be executed along the common execution path;

5
10

5
10

15

15

20

25

30

35

10. The method of claim 9, wherein a multiplexing index is used to multiplex a plurality of virtual channels over a single channel.

5 11. The method of claim 2, further comprising inserting code, which will reformat a message by functions which regenerate constant fields and move non-constant fields to a normal location in the message, into the modified source code.

10 12. The method of claim 2, wherein the modified source code is further modified so that at least one of the common sequence of operations is performed after a message corresponding to the event has been transmitted.

15 13. The method of claim 12, wherein the at least one of the common sequence of operations includes an operation which buffers a message.

20 14. The method of claim 2, wherein the modified source code is further modified so that at least one of the common sequence of operations is performed after a message corresponding to the event has been delivered.

25 15. The method of claim 2, wherein the intermediate data structures are event records designed to pass information between protocol layers in the protocol stack.

30 16. A method for processing an event comprising:

receiving an event at an event processor, the event processor having a trace handler and a protocol stack, the protocol stack having a plurality of protocol
35 layers;

determining whether the event satisfies a trace condition;

5 if the event does not satisfy the trace condition,
 sending the event to the protocol stack and processing
 the event using the protocol layers, the protocol layers
 executing a first kind of operations and a second kind
 of operations;

10 if the event satisfies the trace condition, sending
 the event to the trace handler and processing the event
 using the trace handler, the trace handler executing the
 second kind of operations, but not executing the first
 kind of operations.

15 17. The method of claim 16, wherein the second kind of
 operations include protocol operations.

20 18. The method of claim 17, wherein the protocol
 operations include protocol operations which would be
 executed by the protocol stack if an event which
 satisfies the trace condition were sent to the protocol
 stack.

25 19. The method of claim 16, wherein the first kind of
 operations include the use of event records to pass
 information between protocol layers in the protocol
 stack.

30 20. The method of claim 16, wherein the second kind of
 operations executed by the trace handler are directed by
 computer readable code which has inlined functions, the
 inlined functions being functions which would be called

by the protocol stack if an event satisfying the trace condition were sent to the protocol stack.

21. The method of claim 16, wherein the second kind of operations executed by the trace handler achieve results which would be achieved by the protocol stack if an event which satisfies the trace condition were sent to the protocol stack.

22. The method of claim 16, wherein at least one of the first kind of operations is executed, with respect to a message corresponding to an event which satisfies the trace condition, after the message has been transmitted.

23. The method of claim 22, wherein the at least one of the first kind of operations includes buffering the message.

24. The method of claim 22, wherein at least one of the first kind of operations is executed, with respect to a message corresponding to an event which satisfies the trace condition, after the message has been delivered.

25. The method of claim 16, wherein the second kind of operations executed by the trace handler are directed by computer readable code which has inlined functions, the inlined functions being functions which would be called by the protocol stack if an event satisfying the trace condition were sent to the protocol stack.

26. The method of claim 16, wherein the trace handler executes additional operations which mark a message, the message corresponding to an event which satisfies the trace condition, with a marker which indicates whether

compression operations have been performed in conjunction with processing the event which satisfies the trace condition.

5 27. The method of claim 16, wherein the trace handler executes additional operations which store a reformatting function with a compressed message that corresponds to an event which satisfies the trace condition.

10 28. The method of claim 16, wherein the trace handler executes additional operations which compress an addressing header of a message that corresponds to an event which satisfies the trace condition.

15 29. The method of claim 28, wherein a connection identifier is used to effect compression of the addressing header.

20 30. The method of claim 16, wherein the trace handler executes additional operations which use a multiplexing index to create a virtual channel for sending a message that corresponds to an event which satisfies the trace condition.

25 31. The method of claim 16, wherein the trace handler executes additional operations which compress a constant header of a message that corresponds to an event which satisfies the trace condition.

30 32. The method of claim 16, wherein the first kind of operations include a layering operation.

33. The method of claim 16, further comprising reformatting a message to generate a constant field prior to executing the second kind of operations.

5 34. The method of claim 16, further comprising moving a
non-constant field to a normal location in a message
prior to executing the second kind of operations.

ABSTRACT

Layering of protocols offers several well-known advantages, such as, for example, reduction of a network design complexity, but, on the other hand, layering introduces overhead which increases delays in communication and typically leads to performance inefficiencies. The present invention provides a number of techniques allowing to model protocol layering and detect where performance inefficiencies occur in the stack of protocol layers. Furthermore, after common execution paths are identified in the protocol stacks, these paths are optimized by using optimization techniques, such as optimizing the computation, compressing protocol headers, and delaying processing. All of the optimizations are automated in a compiler with the help of minor annotations by the protocol designer.

20 BFLODOCS:436927_1 (9D4V01)

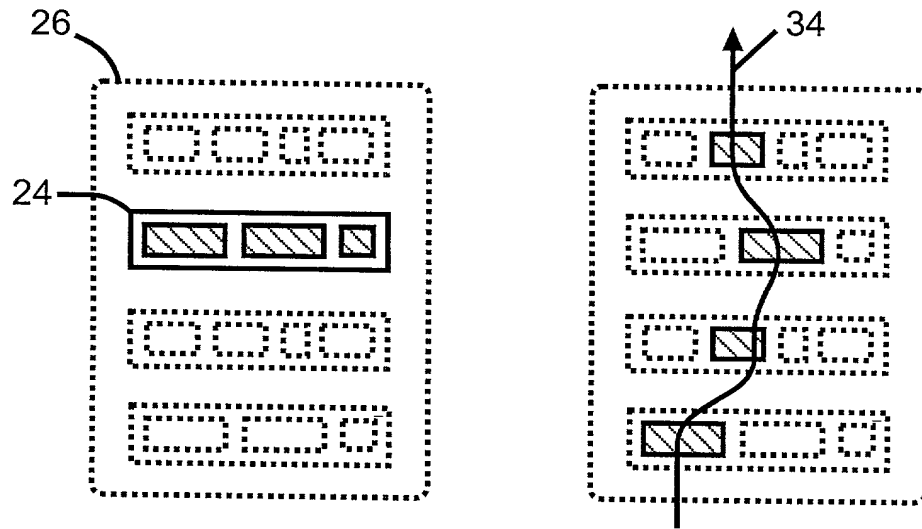


FIG. 1

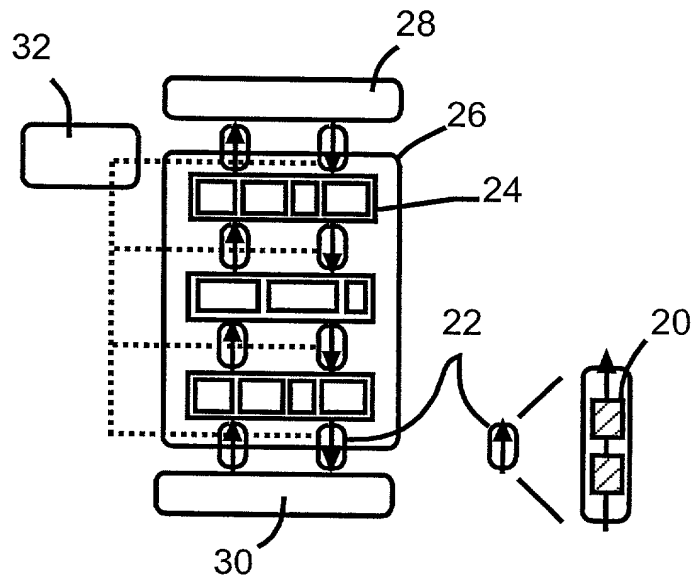


FIG. 2

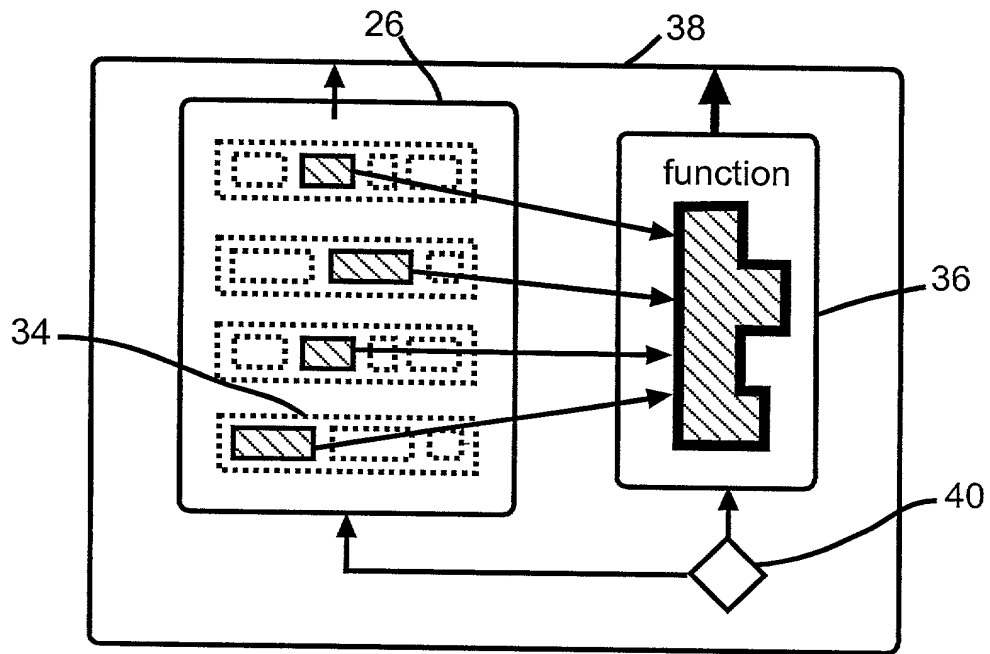


FIG. 3

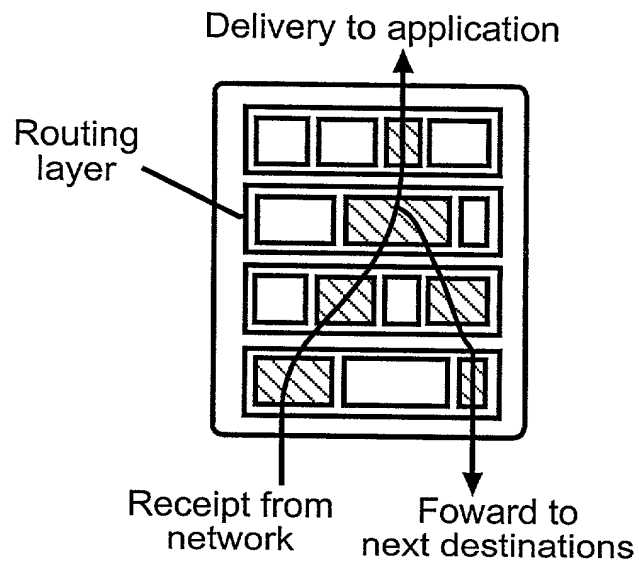
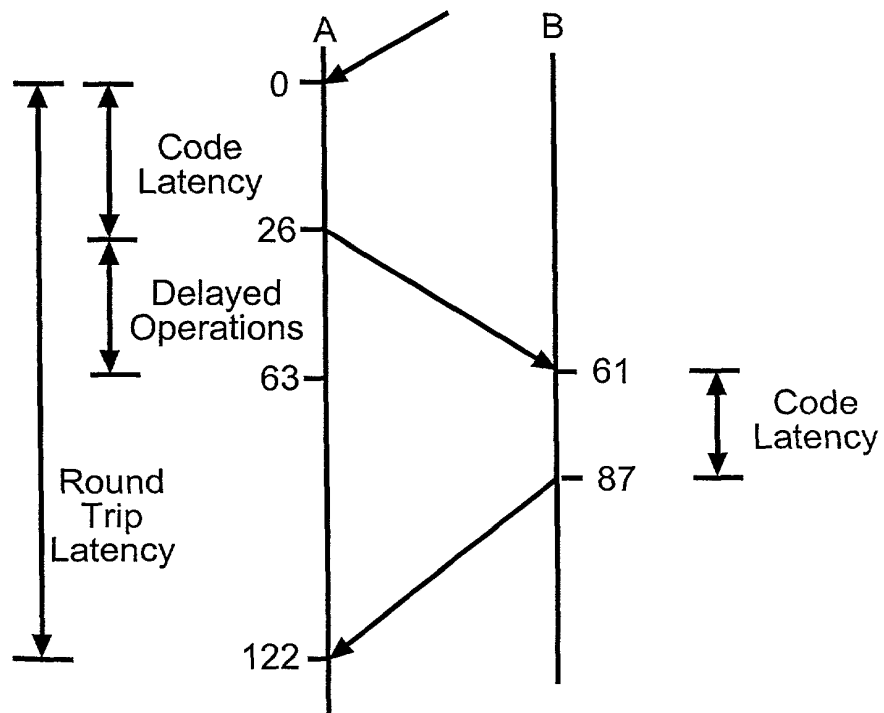


FIG. 4

protocol suite	code latency	delayed operations
normal	1500 μ s	none
trace/ML	41 μ s	28-63 μ s
trace/C	26 μ s	37 μ s

—FIG.5



—FIG.6

8-27-98 ; 3:58PM ;

Attorney's Docket No. 18617.0049**DECLARATION AND POWER OF ATTORNEY - USA PATENT APPLICATION**

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name.

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled: Method and System for Optimizing Layered Communication Protocols; the specification of which was filed on August 28, 1998 by United States Postal Service Express Mail No. EE349081025US

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose to the U.S. Patent and Trademark Office all information known to me to be material to patentability as defined in Title 37, Code of Federal Regulations, §1.56.

I hereby claim the benefit under 35 U.S.C. §119(e) of the U.S. provisional patent application listed below:

Application No.
60/057,602

Filed
August 30, 1997

POWER OF ATTORNEY

I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith.

Maria M. Eliseeva	Reg. No. 43,328
Edwin T. Bean, Jr.	Reg. No. 16,639
John M. Del Vecchio	Reg. No. 42,475
Ranjana Kadle	Reg. No. 40,041
Martin G. Linihan	Reg. No. 24,926
Kevin D. McCarthy	Reg. No. 35,278
Daniel C. Oliverio	Reg. No. 33,435
David L. Principe	Reg. No. 39,336
Michael F. Scalise	Reg. No. 34,920

DECLARATION

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false

006587439-101300
"SECRET"

statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful, false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of sole or first inventor: Robert Van Renesse

Inventor's Signature: 

Date: 8/31/98

Residence: 100 Franklin Street, Ithaca, New York 14850

Citizenship: U.S.A.

Post Office Address: Same as above

Full name of second inventor: Mark Hayden

Inventor's Signature: 

Date: 9/1/98

Residence: ~~111 North Quarry Street, Ithaca, New York 14850~~

Citizenship: U.S.A.

Post Office Address: Same as above

840 MERIDIAN, #103
SAN JOSE, CA 95126

SEND CORRESPONDENCE TO

DIRECT TELEPHONE CALLS TO:

Hodgson, Russ, Andrews, Woods & Goodyear, LLP
1800 one M & T Plaza
Buffalo, NY 14203

Maria M. Eliseeva

(716) 856-4000

SCANNED, #